

## TITLE OF THE INVENTION

### INFORMATION DISTRIBUTION METHOD AND SYSTEM

## BACKGROUND OF THE INVENTION

### Field of the Invention

The present invention is directed to a method and system for integrating heterogeneous information services, and more particularly to a method and system for assembling heterogeneous information streams with asynchronous or digital rates of arrival into a single information stream.

### Discussion of the Background

Real-time delivery of information is an integral part of the decision making process of many brokers and investors. Accordingly, real-time information services spend a considerable amount of money delivering such information to a multitude of subscribers. Generally, such information is distributed over a series of wide-band pipes (e.g., leased lines from the information server). Such a delivery mechanism, however, is contrary to the distributed communication architecture used generally across the Internet. On the Internet, subscribers generally are connected to an Internet Service Provider on a connection shared for multiple purposes (e.g., Web browsing and stock trading).

## SUMMARY OF THE INVENTION

It is an object of the present invention to provide real-time information services without requiring a dedicated, communications medium.

It is a further object of the present invention to provide an event driven information delivery system that combines plural information streams.

It is another object of the present invention to provide an integration of event driven information delivery systems in middleware.

It is yet a further object of the present invention to provide an event driven information delivery system that combines plural information streams into real-time or snapshot delivery methods.

## BRIEF DESCRIPTION OF THE DRAWINGS

A more complete appreciation of the invention and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, wherein:

Figure 1 is a schematic illustration of a computer for providing information integration services according to the present invention;

Figure 2 is a schematic illustration of plural computers acting in concert to provide information integration services to multiple clients according to the present invention;

Figure 3 is a conceptual diagram of plural data clients being provided with data integrated from plural information services using middleware; and

Figure 4 is a block diagram showing the integration of non-financial and financial data sources using middleware;

Figures 5A-5C is an exemplary XML stream for displaying real-time data;

Figures 6A and 6B are block diagrams illustrating integration of middleware with a cache server and other external services to increase the functionality of the present invention;

Figures 7-9 are exemplary screenshots illustrating data display methods according to a first aspect of the present invention;

Figures 10-12 are exemplary screenshots illustrating data display methods according to a second aspect of the present invention;

Figures 13-15 are exemplary screenshots illustrating data display methods according to a third aspect of the present invention;

Figure 16 is an exemplary screenshot of a visual tool for creating, storing and modifying configurations of servers (e.g., priority, data, and login servers); and

Figure 17 is an exemplary screenshot of a visual tool for analyzing the status of a network of clients and servers.

## DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring now to the drawings, wherein like reference numerals designate identical or corresponding parts throughout the several views, Figure 1 is a schematic illustration of a computer system for providing integrated information services. A computer 100 implements the method of the present invention, wherein the computer housing 102 houses a

motherboard 104 which contains a CPU 106, memory 108 (e.g., DRAM, ROM, EPROM, EEPROM, SRAM, SDRAM, and Flash RAM), and other optional special purpose logic devices (e.g., ASICs) or configurable logic devices (e.g., GAL and reprogrammable FPGA). The computer 100 also includes plural input devices, (e.g., a keyboard 122 and mouse 124), and a display card 110 for controlling monitor 120. In addition, the computer system 100 further includes a floppy disk drive 114; other removable media devices (e.g., compact disc 119, tape, and removable magneto-optical media (not shown)); and a hard disk 112, or other fixed, high density media drives, connected using an appropriate device bus (e.g., a SCSI bus, an Enhanced IDE bus, or a Ultra DMA bus). Also connected to the same device bus or another device bus, the computer 100 may additionally include a compact disc reader 118, a compact disc reader/writer unit (not shown) or a compact disc jukebox (not shown). Although compact disc 119 is shown in a CD caddy, the compact disc 119 can be inserted directly into CD-ROM drives which do not require caddies. In addition, a printer (not shown) also provides printed listings of the integrated information from disparate information services.

As stated above, the system includes at least one computer readable medium. Examples of computer readable media are compact discs 119, hard disks 112, floppy disks, tape, magneto-optical disks, PROMs (EPROM, EEPROM, Flash EPROM), DRAM, SRAM, SDRAM, etc. Stored on any one or on a combination of computer readable media, the present invention includes software for controlling both the hardware of the computer 100 and for enabling the computer 100 to interact with a human user. Such software may include, but is not limited to, device drivers, operating systems and user applications, such as development tools. Such computer readable media further includes the computer program product of the present invention for integrating information services. The computer code devices of the present invention can be any interpreted or executable code mechanism, including but not limited to scripts, interpreters, dynamic link libraries, Java classes, and complete executable programs. Computer code devices may also be downloaded across a network using a network adapter (e.g., token ring or Ethernet) as an equivalent to embedding the computer code device within a computer readable medium.

As shown in Figure 2, a variety of content from plural information services are channeled through a series of data servers (e.g., a cluster of NT servers) that coordinate with priority servers. In order to keep pace with the number of users simultaneously requesting

information, the present invention preferably includes redundant data servers, one of which serves one set of users and another which serves other users. The number of connections to each data server is dynamically load-balanced (either at connection time or during connections). In the presence of a failure of one of the data servers, communications routed to a client from one data server are passed off to another data server. The responsibility of reconnecting a client in light of failure of either a data server or a priority can be assigned to any one, or a combination of, the client, the data server and the priority server. Generally, a client would see that services have disappeared when a connection to a priority server is lost, and the client would seek to re-establish communications with another server based on a series of rules on primary and secondary priority servers. Similarly, if a priority server loses contact with a data server, it can seek out a replacement data server based on a series of rules on primary and secondary data servers. (To load balance the servers during operations, lower level servers can periodically shut down all connections to upper level clients/servers such that the connections are re-established using rules that will spread the load. For example, priority servers may periodically shut off connections to all clients such that the clients seek out new priority servers.)

The system may increase capacity by adding more servers (data, priority, etc.). In some cases multiple servers can be configured to run on a single machine. For example, a priority server and a login server can run on a single computer, but there must be at least one other computer running both server applications. In fact, as shown in Figure 3, the services of plural servers can be integrated into a single middleware product to facilitate its incorporation into third party products.

The information from the data servers can include information from stock exchanges, futures markets, news organizations, and data analysis feeds. The information is delivered via a multi-directional, wide area communication protocol (e.g., UDP/IP, RDP/IP, TCP/IP).

Generally, for some types of information error detection and retransmission is necessary, but for other types of data, it is sufficient to wait until the next update arrives to provide the user with data that was lost or corrupted. In the context of an error-correcting protocol, the lower level servers may communicate change information only since a client is guaranteed to have the most recent information. In non-retransmission systems, it is preferable to send the absolute value of the information rather than just the change.

An alternative protocol, PCT/IP, designed by applicants provides several benefits over raw TCP/IP. It initiates connections between client and server. It provides the option of having any number of primary servers or nebula to connect to. It also provides the option of having any number of backup servers to connect to in the event the primary are unreachable. It will automatically reinitiate connections allowing for failover capability. It provides compression and encryption for transmission of data.

It is possible to transfer data by parsing the data fields sequentially into a message for transmission and then parse the data fields out on the receiving end, but such a process may be cumbersome. Instead, data structures may be transferred directly by encoding objects into a stream of bytes allowing for greater compression since there is no parsing information contained in the message. This may increase speed since the message is smaller and there is less processing involved in the transmission and reception.

Once in the servers, the data are post-processed (or "normalized" into a standard format) and passed to other servers and clients via an Application Program Interface (API). In addition to data retrieval through the API, data can be collected using other communications mechanisms and protocols (e.g., OLE, DDE, ActiveX, Webula, and XML).

As shown in Figure 4, the data is not restricted to financial data. Any event may be added to an integrated data stream by processing the data through the API that normalizes and transmits data. Non-financial data includes order entry and fulfillment, inventory control, production system, pipeline flow measurements, tracking of train cars, busses, trucks and automobiles, refinery monitoring, warehousing, oil and gas extraction, mining and quarrying, paper manufacturing, defense and military, printing and publication, e-commerce and e-mail data processing, measuring analyzing and instrumentation, waste management and remediation, education services, health care and social assistance, arts, entertainment, and recreation, accommodation and food services, etc.

Delivery of snapshot information from priority servers to a data client (e.g., on the Web) occurs via dynamically generated content (e.g., active server pages (ASP)) as the client requests data. In a Web page embodiment, the priority servers can further format the normalized data using scripting languages (e.g., JavaScript or VB Script). Such data can be included in frames, tables, and all other standard browser structures. Using HTML codes, data can be displayed multiple colors and fonts. Snapshots work well with browsers (e.g., Netscape and Internet Explorer).

Streaming delivery is sometimes called event driven delivery. Simply, it means that any change in any data seen by a priority server is streamed ("pushed") to all appropriate users on their displays (e.g., browsers). Where the updated information is displayed on the native web page is based on the location as authored in the dynamic display (e.g., web page). However, by incorporating ActiveX controls in a visual display, the data can be visually updated on a portion of the client display without having to re-request the page in its entirety. Such changes can be displayed in changing colors to accentuate the changes. Moreover, for multi-field displays, it is preferable to allow customization of the order of those fields. For example, the order and/or presence of fields related to the contracts (e.g., Last, Open, High, Low) are individually selectable.

In one embodiment of the present invention, the streaming data is provided in an XML format, as shown in an exemplary format of Figures 5A-5C. By utilizing the XML format, data objects and their attributes within the stream can be quickly identified and displayed. The XML format need not be in plain text, however. In one embodiment, to further reduce bandwidth, the information sent from a priority server to a client is sent in binary, compressed, or abbreviated form. Using the information stored in the priority servers (and received from the clients), the priority servers know the events (e.g., stock price changes or order entry completion) being tracked by the clients. Thus, individual events can be given shorter representations that are converted into streaming XML on the client-side.

Event driven data delivery in a Web browser is complicated by the lack of browser standardization. Currently Internet Explorer handles event driven data far better than Netscape. Accordingly, an ActiveX control is used as a "least common denominator" that is small and installed quickly. Future browser improvements will allow similar functionality on Netscape and other browsers using plugins, applets, com objects, and other controls.

To obtain permission to log into the Digital Integrator servers a user must complete an enrollment process by filling out online forms. During this process the user's personal information such as name, address, e-mail and credit card is obtained through a proprietary system created by Digital Integrator. Also selectable in this process are the services to which the users wish to subscribe (specific news services, specific quote feeds, etc.). After the user's data is captured the user will be able to select a unique user name and password combination that allows access to precisely the information to which the user subscribed. (Preferably users receive their unique user names from a series of web pages that collect customer billing and

demographic information.) The permissioning can be extremely precise, allowing access to the smallest subset of a feed or it can be very broad, allowing the grouping of many feeds. This is an in-house e-commerce system developed by Digital Integrator for use with the middleware.

The initial connection from a client program to the servers of the system is via Login Servers. Login Servers validate client user IDs, passwords, and data service permissions via Authentication Servers. Once permissions are established, the client is routed through the Priority Servers to the List Server/Data Server combination that has the least traffic. Only the data requested and permissioned by the client is sent through the network, minimizing network traffic. Information to a front-end program from the Data Servers is routed via the most efficient path with the lightest load.

Data Servers are optimized to speed data delivery. Different types of data servers are customized to handle the unique characteristics of each data feed. Record-based quote data servers are different than fixed page data servers or news/text data servers, but they are all normalized to allow a standardized output to a single client application. The distributed processing architecture allows individual components, described below, to handle administrative and maintenance work maximizing throughput from data servers to clients. Specialized database servers add the ability to archive and retrieve historical data (e.g., using a SQL Server system).

Moreover, as shown in Figures 6A and 6B, speed may be further increased using cache servers that connect to middleware or equivalent separate servers. The cache servers allow the caching and instant throughput of data on servers that reside on the client computers and/or on local networks that receive data from the middleware for a group of users on the receiving local networks. Caching servers can also handle more complex tasks like the translation of protocols. In one embodiment of an environment in which several related or co-located clients wish to receive the same non-real-time information, the several clients first attempt to retrieve the information from the cache server before requesting the data from the priority server. (In an alternate embodiment, the real-time data passes through the cache servers as well.)

The Login Servers handle the primary load balancing between the clients and the data servers. They direct traffic to the servers best able to handle the load (e.g., the lightest loaded or the one already serving co-located clients). Before a connection is established Authentication Servers verify user ID and passwords. Priority servers rank the order (priority)

of the delivery of different types of data. Real-time data will always have the highest priority. Initial downloads to populate the screen with current data and requests for historical data will have lower priority. This functionality of the priority servers insures that the most time-sensitive data will get to the end user first. Data from multiple data servers are combined in priority servers and sent through to the client or cache server using a single connection. A final function of the priority servers is that they ensure that data goes only to permissioned users.

Configuration servers constantly monitor the configuration status of all servers. This service shares with priority servers the self-healing intelligence to instantly reroute all connections to accommodate an overloaded or failed server.

List servers keep track of the quote data being requested by the Clients. Only the issues that are in active client windows are updated in real-time. List server functionality is built into some of the data servers, but it could be built into other servers such as the priority servers.

In light of the nature and importance of the real-time data, the present invention preferably withstands the failure of a single computer or a bank of computers with minimal degradation in the delivery of data. One factor in withstanding failures is the ability to use redundancy to perform self-healing of the system. Preferably the redundancy includes server farms in physically remote locations (e.g., New York, New Jersey and the Western US). Additional levels of redundancy can span countries (e.g., having server farms in the U.S., London, and Hong Kong. Each such redundant server farm acts as a backup for the others such that all connections can be re-established through at least one redundant server farm.

Automatic failover transfer from one server farm to another is accomplished in one of three ways. First, a transfer may be initiated from the server-side. All servers and server farms are constantly monitored by a monitoring system. Technicians are alerted when a problem arises in a single server or a bank of servers. Sound, visual, and telephony alarms may be used to alert technicians when anything goes wrong with a data feed, a software service, or a hardware function. The monitor tells the exact nature of the problem on an individual server basis. When no one is physically available at the server farm the monitoring system automatically dials and gives exact messages to an off-site technician. Based on the assessment of at least one technician, the technician(s) can force one or more clients to transfer from one server farm to another. As a separate server-side transfer, the system may automatically force



one or more clients to transfer from one server farm to another based on the type of problem encountered.

Second, a client program may automatically request a connectivity transfer between server farms without user intervention. For example, a client program may constantly monitor the status of its connection to a server farm. When the connection lapses for more than 30 seconds, it is automatically rerouted to a healthy server farm. Moreover, programs may share a dynamic linked library that reroutes connections to avoid recoding of the reconnection feature for add-in clients.

Finally, an end-user can manually direct, or redirect, the connection to any server farm. If the user detects a connection problem before the connection has been automatically redirected, the user has the power to force a connection to a different system. This feature is also of interest to customers who travel to remote locations. For example, a customer who travels to Australia from New York will want to direct his laptop computer to connect to the Sydney server farm instead of the New York server farm.

In addition to the server hardware and software, raw data feeds can be directed to each server farm. Equipment to handle all incoming data feeds and outgoing delivery are installed and configured at each site based on required connectivity and/or bandwidth.

In order to facilitate customizable information presentation, the present invention includes configurable screens that are arranged according to a user's preferences. Users can create their own pages with windows full of diverse data from all of the data sources that are being integrated by the system. Preferably, within the client interface, (1) historical data is easily retrievable, (2) quote data can be charted and graphed in custom formats, (3) news can be searched and profiled, (4) Web page windows can be seamlessly integrated with quote, news and analysis windows, and (5) collections of windows can be easily combined and saved as user designed "pages".

Figure 7 illustrates an exemplary interface according to a first aspect of the invention. The interface (taken from the program entitled "PC Trader" owned by the assignee of this application) is a product that delivers real-time financial data across a wide area network (e.g., the Internet or any TCP/IP network) from a middleware (e.g., the middleware layer known as the "Digital Integrator"). Such a visual interface may be written in any programming language (e.g., C++, Visual Basic, FoxPro, and Java), but the illustrated version is written in C++. The data may further be communicated using a protocol other than TCP/IP (e.g., the PCT/IP

protocol discussed above). Such an interface may be configured to run on a workstation, a stand-alone computer, or any other computer system. According to the present invention, the operating system is a matter of design choice, and equivalent systems can be run in top of Windows 95/98/NT/2000/ME, Unix, Linux, etc.

As illustrated in Figure 8, in the preferred embodiment the user's real-time analysis is supplemented with a charting function that utilizes the real-time data that is received. Various objects work in cooperation to form the illustrated interface that simultaneously updates and displays quotes from the cash treasury markets, the futures markets, and news. One object of Figure 8 (labeled "Treasury Composite") illustrates an updating page of financial data from an older legacy architecture. This type of screen object updates itself using commands to write at particular screen coordinates corresponding to the part of the page that has changed. The other type of screen objects are updated using "records". Each symbol (e.g., stock, index, futures contract, etc.) is a record that has unique identifiers. Both record-based and page-based objects can be updated in real-time. Moreover, the updates may include supplemental multimedia streams (e.g., sounds and graphics).

As illustrated in Figure 9, other object types can performed more advanced calculations on the real-time data before displaying it. The illustrated real-time chart displays analytic information after processing (e.g., using arithmetic, logic and rule-based algorithms) the primary data received from the servers/middleware. Generally, the received real-time data can be used to create derived, or secondary, real-time data.

As illustrated in Figure 10, a dedicated user interface (e.g., PC Trader) is not required to receive and process real-time data. Instead, a web browser may also receive real-time updates from the Middleware or a remote server. The illustrated interface receives (1) record-based quote information from the Chicago Board of Trade (CBOT) and (2) news from the CBOT or another source. The quote data is updated periodically (e.g., every ten seconds) directly from the Middleware. In a preferred embodiment, the data is pushed directly from the middleware to the browser. In an alternate embodiment, the data is refreshed (i.e., pulled). Having selected the "Stock Index" tab at the top of the illustrated page, the stock indices are updated in an HTML table. In a preferred embodiment, the browser on the client's computer receives the data in the binary PCT/IP format, then translates the data into a streaming XML format that is then read by the browser. In an alternate embodiment requiring additional bandwidth, the data is sent directly from the Middleware in pre-formatted XML.

Figure 11 illustrates sub-areas within the "Combinations" area. Other tabs on the top of the browser page provide quick access to various pre-set choices that the user can select. As with the other embodiments described above, the users receive live data only after logging into the Middleware (e.g., using a verifiable user name and password); however, once logged in, the user can switch from page-to-page without re-authentication. Similar to the interface of Figure 11, Figure 12 illustrates sub-areas within the "Financials" area.

Although shown as static layouts in Figures 10-12, the system administrator may also enable an alternate interface that enables customization. However, due to management reasons (e.g., the additional technical support required with a customizable interface), the type of interface used by customers may be restricted by an administrator on a customer-by-customer basis.

Figure 13 illustrates an alternate Web interface that receives its updates from a middleware layer. In the illustrated embodiment, each of the different type of quote or news window is contained in a layer (window). Those layers can contain any type of HTML object that in turn can display the XML data that is derived from the middleware layer. The layers can be moved and resized by the user. The frame on the right shows the customization interface. The end user can completely customize, create and delete the data in each of the layers.

Figure 14 illustrates yet another alternate interface within a browser. The top of the screen shows a username and password login interface. The customer's login information is kept in a relational database that is referenced every time a user logs in. This database specifies the type of data that the user is allowed to access (e.g., quote types, news sources, level of service).

Figure 15 illustrates a Web page that groups the data window-like display objects using frames instead of layers. However, as would be understood by one of ordinary skill in the art, other types of display objects (windows) are possible. For example, instead of using window-like objects that are layers as shown in Figure 13 or frames as shown in Figure 15, multiple "child" browser windows (essentially stripped down versions of Internet Explorer or Netscape) can be managed by a single "parent" browser page. These child windows can be created and edited by the end user to contain any type of custom display that is important to that user. As with all types of customized display objects, the size and configuration of all of the child windows can be stored in a cookie or configuration database. After the data is

stored, the user's customized settings will automatically appear on the screen the next time the user goes to that page.

In order to reduce the bandwidth utilized by communication, multicasting is used to send repetitive packets to multiple sites at the same time. When real-time packets arrive for retransmission, via a feed from a data vendor, a private source of information, or data customer of IXnet, they are substantially instantly processed in multiple data servers and sent to all clients who have permissions to receive the packets. Because of the fundamental distributed processing foundation of Nebula, it has the ability to "multicast" to an enormous number of clients.

When a new news story is sent on a news wire the system instantly multicasts only the headline to all of the clients with permissions to view that news wire. The story text is sent to clients when they request a specific story. This represents a tremendous reduction of bandwidth usage. Moreover, the headlines may be further reduced using filtering of stories based on key words and other search criteria within headline windows and "smart story" windows.

The multicasting techniques built into the quote servers utilize the services of the list and priority servers. This combination of servers keeps track of the data actually being requested by all of the client programs (e.g., thousands) that are connected to the server farm. When quote packets for a specific quote symbol are received by the server farm, they are instantly multicast to all clients actually monitoring that symbol. Because all quote symbols are not always sent, this multicasting technique massively reduces network traffic and increases quote delivery speed.

In order for that selective distribution technique to work, however, the client programs must track all quote symbols that are included on all quote windows (minimized, normal, or maximized) on the currently active page, and communicate those symbols to the priority and/or list servers. The list servers keep track of the ever-changing list of requests from all active clients, and the requests from the lists are multicast in real-time.

For large installations on a single local area network, Internet and extranet traffic can be minimized by the installation of an on-site multicast server. This server communicates with all of the clients on the local area network and with the servers in server farm and reduces network traffic by delivering a local multicast.

The following example illustrates the local multicast functions. The multicast server will keep track of the fact that 150 local clients are viewing real-time news, 55 clients are viewing the 10-year Treasury note, and 75 clients are viewing the long bond contract from the Chicago Board of Trade. The local multicast server will request and receive only one set of packets from the server farm when each of the items are updated, but it will instantly pass through the updates to all of the local clients requesting the data. This represents a measurable savings in bandwidth which in turn lowers delivery costs to clients.

As discussed above, remote control and administration of server farms leads to increased responsiveness to problems. Using a remote management system, technicians can reboot servers, load new software, manually reconfigure the system, move users from one server to another, terminate users, and make other adjustments from office or home. The system maintains a central database of all configurations of all Digital Integrator middleware servers and the connections to other servers. Users of the middleware can dynamically reconfigure any nebula or server remotely to accommodate server failure or data feed loss or similar situations allowing for greater redundancy and failover protection. The system can either (1) maintain exception statuses manually or (2) intelligently monitor server statuses and reconfigure the middleware appropriately when the automatic failover protection within a middleware server farm is insufficient.

Figure 16 illustrates a visual tool for creating, storing and modifying configurations of servers. By selecting two servers and then the "connect" option, an administrator signals that a connection between the servers is to be generated. The connection can be generated either (1) the next time that the servers read a resulting configuration file or (2) immediately by sending a message to at least one of the affected servers.

Figure 17 illustrates a visual tool for displaying the status of each of the servers in a selected group of servers within at least one network. The color-coded legend on the right-hand side of the tool's display indicates the status of each of the servers. Using this tool, an administrator can be selectively called by setting the "midnight caller" option to enabled. Using that option, a series of numbers are dialed in order to find a person that can service the error detected by the system. The notification is preferably made using a text-to-speech converter, but other notification methods (e.g., e-mails, pages, and wireless messages) can also be used.

Obviously, numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that, within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.